Computers and Mathematics with Applications 63 (2012) 544–553

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Versatile iPad forensic acquisition using the Apple Camera Connection Kit

Luis Gómez-Miralles, Joan Arnedo-Moreno *

Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya, Carrer Roc Boronat 117, 08018 Barcelona, Spain

ARTICLE INFO

Keywords:
Forensics
IPad
Cybercrime
Digital investigation
Apple

ABSTRACT

The Apple iPad is a popular tablet device presented by Apple in early 2010. The idiosyncracies of this new portable device and the kind of data it may store open new opportunities in the field of computer forensics. Given that its design, both internal and external, is very similar to the iPhone, the current easiest way to obtain a forensic image is to install an SSH server and some tools, dump its internal storage and transfer it to a remote host via wireless networking. This approach may require up to 20 hours. In this paper, we present a novel approach that takes advantage of an undocumented feature so that it is possible to use a cheap iPad accessory, the Camera Connection Kit, to image the disk to an external hard drive attached via USB connection, greatly reducing the required time.

1. Introduction

Portable devices have become a very important technology in society, allowing access to computing resources or services in an ubiquitous manner. In this regard, mobile phones have become the clear spearhead, undergoing a great transformation in recent years, slowly becoming small computers that can be conveniently carried in our pockets and managed with one hand. However, as user requirements started to include new functionalities beyond those that a mobile phone can realistically offer, advanced portable devices have been developed in order to fulfill them. Such devices try to reach a compromise between a high degree of portability, usability and the ability to provide such advanced functionalities (for example, being able to read or process documents).

One of the top devices in the field of embedded portable devices is the Apple iPad, a tablet computer which tries to take advantage of the success of its ancestor, the iPhone. It was announced by Apple in January 2010 and launched in the USA and Europe between April and May 2010. After 80 days in the market, 3 million units had been sold [1]. Given its popularity, it becomes evident that as such devices become widespread, they will also become more common and relevant as sources of evidence from a computer forensic standpoint, providing data about their users. This kind of data can become very important in cases of criminal investigation, where it can be used as evidence in court or provide valuable clues to investigators. Since advanced portable devices are usually closed embedded systems with their own idiosyncracies, not actually being full-fledged PCs, forensic data acquisition presents some interesting challenges. This is especially relevant when it is necessary to use non-invasive methods, maintaining the device in the same state (or as similar as possible) as the one it was in before the analysis began.

Currently, the easiest method to obtain a forensic image of an iPad device (which can also basically be applied to an iPhone) is to install an SSH server and some tools, retrieve its internal storage contents and transfer the data to a remote host via wireless networking. Unfortunately, this is very slow, and can take up to 20 h. More efficient methods exist, but they

* Corresponding author.
E-mail addresses: pope@uoc.edu (L. Gómez-Miralles), jarnedo@uoc.edu (J. Arnedo-Moreno).

0898-1221/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2011.09.053

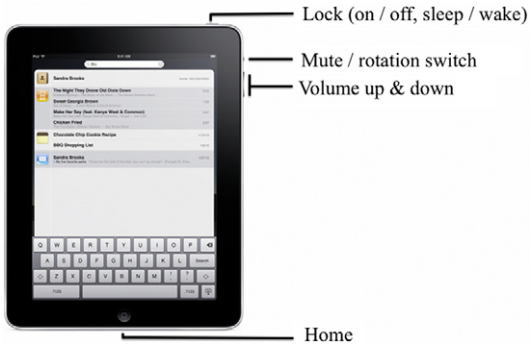

Fig. 1. iPad button configuration (iOS 4 and higher).

are based on closed software, which relies on obscure or undisclosed techniques and is very dependent on the iOS version. As soon as an iOS upgrade becomes available, they can no longer be used until a new version is created.

In this paper, we propose a versatile approach which reaches a compromise between forensic data acquistion speed and an open approach, without the need of vendor specific software or a dependency on a very specific iOS version. Achieved with the help of a cheap and easily available peripheral, the Camera Connection Kit, it is possible to generate a forensic image using an USB connection. Furthermore, this approach still minimizes the amount of data which is modified during the acquisition process.

The paper is structured as follows. Section 2 provides an overview of the iPad architecture, focusing on those characteristics especially relevant from a forensic analysis standpoint. In Section 3, a literature review of the current state of iPhone/iPad forensic imaging techniques is presented. The proposed forensic data acquisition method is described in Section 4. Section 5 provides a brief discussion about which forensic techniques can be applied on the image generated using our method, and, in Section 6, an analysis of its performance is presented. Concluding the paper, Section 7 summarizes the paper's contributions and outlines further work.

2. iPad architecture overview

From the external point of view, the iPad is basically a big (24 × 19 cm.) iPhone with a 9.7″ screen, providing a resolution of 1024 × 768. While its internals are very similar to those of its ancestor, the iPad's larger form factor makes it suitable for longer periods of use, which has motivated the appearance of many different applications of all kinds. Therefore, the iPad is able to perform tasks previously reserved for common computers or, up to some point, netbooks.

2.1. Main features

The basic iPad internals are:

- Processor: A custom Apple *A*4 ARM processor based on a single-core Cortex-A8, running at 1 GHz.
- Volatile storage: 256 MB DRAM.
- Non-volatile storage: 16, 32 or 64 GB solid state storage drive.
- Wireless connectivity: 802.11 a/b/g/n and Bluetooth 2.1, the same as every iPhone.
- In addition, the 3G model features an A-GPS (Assisted GPS), and hardware for communicating over UMTS/HSDPA (820, 1900 and 2100 MHz) and GSM/EDGE (850, 900, 1800 and 1900 MHz).

A second generation hit the market in March 2011, featuring a dual-core Apple *A*5 processor and 512 MB of RAM.

2.2. Connectors and buttons

The iPad connectors and buttons are very similar to the iPhone's. When placed over the short edge with the round button in the center, we find:

- Top left: a 3.5″ jack capable of functioning simultaneously for several audio functionalities.
- Top right: "Lock" button.
- Right edge, near the top: volume controls, and one configurable switch which can either mute the device, or lock the screen orientation.
- Bottom center, front face: round "Home" button.
- Bottom center, in the edge (below the "Home" button): Apple standard 30-pin "Dock" connector, the same as used in every iPhone and most iPods. This is also the port used when referring to the iPhone or iPad's 'serial port'.

Fig. 1 shows the function of each button. Note that the "Lock" button performs several functions: when the device is off, it will turn it on; when the device is on, a short press will put the device to sleep or wake it from sleep and a long

press will show a dialog to turn it off. For clarity, in this paper we will refer to this button as the "Lock" button. The button configuration is important since, as will be explained in Section 4.1.1, it may be necessary to put the device in *DFU mode* ('Download Firmware Update') in order to setup the device for forensic imaging. When this is needed, installed software usually instructs the user to press a particular combination of these buttons to have the device enter DFU mode.

2.3. Partition scheme

As noted by Zdziarski [2], all devices belonging to the iPhone family contain two partitions:

(1) A huge *user data* partition, holding all extra installed applications as well as all the user's data.
(2) A small *system* partition containing iOS and the basic applications.

From a forensic standpoint, as far as the user data partition is concerned, some iPad applications that may hold relevant data include enterprise or office software, such as QuickOffice Connect Mobile Suite [3] or Apple's iWork suite [4]. They can all contain text documents or spreadsheets, which are prone to including sensitive or financial information. Although similar applications existed in the iPhone, allowing for direct document editing without the need for an external computer, the iPad's form factor will no doubt boost the existence of documents stored solely within the device (and not, for instance, in the suspect's main computer), being edited here and never moving beyond the iPad (with the possible exception of device backups performed by iTunes).

Another possible source of information lies within Apple's *AirPrint* framework, released in November 2010 as a feature of iOS 4.2 [5], which provides native printing capabilities to the iPhone and iPad. However, long before AirPrint existed, other applications, such as PrintCentral [6], already allowed the user to send most document types to a remote printer (connected to a computer with the appropriate server software). These applications' disk caches are likely to hold relevant information, such as copies of printed documents.

The system partition contains the base iOS software that comes bundled inside every iOS software update (which explains why they require hundreds of megabytes). This includes the core operating system and graphical user interface, as well as the standard set of bundled applications such as: Safari, Mail, Calendar, iPod, etc. Note that only the application binaries themselves lie within this partition, whereas the relevant data (for instance, user mail) is stored in the data partition.

3. Current work on iPad forensics

A very basic approach to acquiring user data is to connect the device via the standard USB cable to a computer running iTunes, Apple's multimedia player, which is in charge of synchronizing content to the device. Using its AFC protocol (*Apple File Connect*), iTunes syncs existing information (contacts, calendar, email accounts, apps…) and can even retrieve a complete backup of the device; however this presents two problems:

(1) The device needs to be correctly paired with the iTunes software in order to sync.
(2) Even if the investigator has access to an iTunes backup of the device (say, found in the suspect's main computer), it will not contain unallocated space, from which deleted data can be recovered.

Consequently, more sophisticated methods are required. However, the iPad is distributed as a closed device, meaning that access to its internals is limited and only applications approved by Apple may be installed or executed. With this set of restrictions in place, it is extremely difficult to acquire any kind of meaningful forensic data. Fortunately, even though the iPad is a very new device, its internal architecture is very similar to the iPhone and forensic approaches may be easily ported to the iPad.

On July 6th 2007, just one week after the iPhone was launched, George Hotz announced [7] the existence of a method to provide a full, interactive shell. This was the first step towards bypassing Apple's restrictions on their devices, making it possible to execute any program and not only those approved by Apple; a process that has been named *jailbreaking*. In other mobile platforms, such as those running Google's *Android* operating system, a similar process exists, known as *rooting*. Vendors usually disapprove of this technique, although in most countries it is legal or at least not definitely illegal. If you ever need to defend this practice in court, you can give a brief explanation of why you are jailbreaking the device: to get full access to the system, and thus to the information stored in it, which is crucial in criminal cases which require forensic analysis of these kind of devices.

The jailbreaking process modifies the system partition without altering the data partition, which means that it does not alter the user's data, a very important requisite. Even if we assume that some current or future jailbreaking methods will modify the user data partition, we can still obtain plenty of useful information, as long as we know what alterations we are responsible for. Ever since their development, the jailbreaking tools have been updated to support every new iPhone model and every new iOS version. This method may also be applied to an iPad and, in fact, the two major forensic approaches to recovering a complete image from the device are ultimately based on jailbreaking.

The main approach was proposed by Zdziarski [2], who noted that *the iPhone can communicate across several different media, including the serial port*, 802.11 *Wi-Fi, and Bluetooth. Due to the limitations of Bluetooth on the iPhone, the two preferred methods are via the serial port and Wi-Fi*. He proposed a basic method for obtaining a forensic image of the iPhone, without tampering with the user data partition, by jailbreaking the device and using SSH access and the `dd` and `netcat` standard UNIX tools, which by that time had already been ported as a part of the growing iPhone jailbreaking community.

Similar methods are explored by Rabaiotti [8] against a Microsoft Xbox. There was not, however, a known, public way to communicate with the device via its serial port, so Zdziarski had to send the forensic image via the device Wi-Fi interface, which is quite slow.

Alternate approaches are provided by some forensics software vendors [9,10], who have developed solutions that use rather uncommon techniques to get a dump of the solid state storage drive. This is often accomplished by using exploits against more or less known bugs in specific iOS versions in order to execute arbitrary unapproved code, which is actually the same as *jailbreakers* do in order to free their devices. However, these vendors do not need to install a complete set of tools in the device. Instead, they tend to upload a tiny, small-footprint software agent which will ideally take control of the system, dump the solid state storage drive through the serial port (dock connector), and then reboot the device without copying any data to the iPad internal storage. This method offers some advantages over the jailbreaking approach, being a more straightforward process, simpler for the investigator and leaving little or no footprint on the acquired system. However, it also has some weak points.

First and foremost, any proprietary method ultimately makes use of an exploit against the vulnerabilities of the iOS version of the device, because this is the only way to take such control of the device, bypassing every vendor restriction. With every iOS update (usually every few months, downloaded via iTunes), the forensics software must be updated, usually because bugs exploited in previous versions are fixed in the newer version; but even if an exploit still works, exploitation parameters such as memory addresses are very likely to change.

Jansen [11] identified *"the latency in coverage of newly available phone models by forensic tools"* as one of the problems for forensic specialists working with mobile devices. Jailbreaking in the iPad has been moving in a timeframe of barely 1–5 days following iOS updates. We consider it very realistic that at some point in the near future, jailbreak updates will be available days or even weeks before some particular forensics software products get the same needed updates. Therefore, even though some of the proprietary methods may be suitable for the analysis of the device under common circumstances, vendors of such products may fail to release an update in time to support newer iOS versions; they may even not release it at all if, say, the product is discontinued.

In addition, many of these proprietary methods are closed and lack any public documentation. Therefore, they are difficult to audit and it cannot be guaranteed that no footprint is actually left on the device. Knowing the process the device is going through, and the precise alterations that this process causes to the device, is good practice and very important to the forensic investigator.

4. Versatile forensic data acquisition

In this section we will describe the proposed method for iPad imaging via the Camera Connection Kit through the USB connection. The method is divided into two general phases: device setup and imaging. Each phase is also divided into several substeps which must be sequentially followed.

4.1. Device setup

This phase encompasses all the necessary steps to leave the device open and ready for forensic image acquisition. As mentioned in Section 3, before any forensic analysis may be attempted, a special device setup is required in order to bypass the access restrictions installed by the manufacturer. Once this phase is complete, low-level access to the device is actually possible. In addition, it is necessary to install the extra packages needed for our proposed imaging approach.

4.1.1. Jailbreak the device

The only requisite in order to apply our method is starting with a jailbroken iPad. We will not provide detailed instructions about how to jailbreak an iPad, just the basic guidelines. Once this prerequisite is met, our method can be applied on any iPad, iOS version notwithstanding. This is in contrast with vendor specific tools, as explained in Section 3.

The actual way to perform jailbreaking varies depending on the iOS version installed on the device. An iPad running iOS 3.2.1 can be jailbroken just by browsing the website http://www.jailbreakme.com, which exploits a known vulnerability in Safari to take control of the system. The exploits themselves and related documentation can be found at [12]. For a complete, up-to-date chart about jailbreaking tools for each iOS version, refer to [13].

Should we find a device with a recent iOS version for which no jailbreak procedure exists, it could be acceptable to downgrade to the latest jailbreakable version, although this should be done only as a last resort, and always documenting the steps taken. This rarely succeeds, however, because Apple does not allow one to downgrade a device's iOS version after a newer version has been available for some time. There are some workarounds for this but they are of no use in our scenario because they require that we have previously saved some crucial data before installing the present iOS version. Anyway, it is very unlikely that we will hit a non-jailbreakable iOS. Take, for instance, iOS 4.2.1 (the first iOS 4 release for the iPad): it was released on November 22 2010, and the appropriate tool for jailbreaking (in that case *redsn0w*) was released the next day [14]. Similar time windows apply for the different 4.3.x iOS versions released during the first months of 2011.

Once a new iOS version has been released, the first jailbreaking methods will probably be *tethered*: a tethered jailbreak means that it is only effective as long as the operating system is running. The moment it is rebooted (not when the device

is locked), the jailbreak is lost, meaning that two things will happen temporarily until the device is rebooted again into a tethered jailbreak state with the appropriate tool: (1) any jailbreak software installed will not work; and (2) some internal applications (for instance the Safari web browser) may not work, or in the worst case, the whole device might not work at all. We state again that this is only a temporary state, until the device is jailbroken again. It would be acceptable to use a tethered jailbreak for imaging purposes, and in fact part of the tests performed in this paper have taken place on an iPad with a tethered jailbreak.

Many jailbreaking tools (`redsn0w`, `PwnageTool`, etc.) will require the user to put the device into *DFU mode* with a combination of presses of the "Lock" and "Home" buttons. When this is needed, the software will give the user the necessary instructions.

It also is important to note that *jailbreaking* a device does not mean *carrier unlocking* it. Jailbreak is just a precondition for carrier unlocking. Our proposal need not perform carrier unlocking, and in fact this is rarely needed in the iPad given that it is usually sold carrier-free.

4.1.2. Install required software packages

After the device has been jailbroken, a new application labeled *Cydia* [15] appears in the home screen. In the case that this application already exists, the previous setup is not necessary. This is the software manager that allows the installation of software not approved by Apple.

When run for the first time, Cydia initializes the device's filesystem and exits. In the next execution, it presents a *Who are you*? prompt, offering three choices; 'Developer (no filters)' must be chosen, as it offers the widest range of software. Afterwards, if there are available updates to install, it is recommended to perform a 'Complete upgrade'. The device will then restart. It is necessary to re-open Cydia and, if asked for upgrades, repeat the process.

Once Cydia has finished upgrading itself, using the 'Search' function, the following packages are installed:

- *openssh*. This package contains the SSH server that will be used to access the device.
- *coreutils*. This package contains the `split` command, which is needed due to reasons that will be explained later.

The most important tool for this procedure, `dd`, need not be installed, as it is contained inside the essential *coreutils-bin* package, which is installed by default as part of the jailbreaking process.

4.1.3. Network and auto-lock settings

It is necessary to connect the device to a wireless network, since during the process it will be remotely accessed from another computer. This connection is necessary to issue commands, but not for the actual image data transfer. If no wireless network is available, a laptop can be used to create an ad-hoc network and have the iPad join it.

Communication between the computer and the iPad will be over SSH, and thus, encrypted. However, it is strongly advised to use encryption in the wireless network protocol, and ideally, to use an isolated network for the computer and the iPad only. The main reason for this is the fact that there is a small window of time during which the device will be accessible with default passwords. There is at least one known worm which penetrates jailbroken iOS devices using these default credentials [16], although nowadays it is almost impossible to find that code in the wild.

Once the connection has been established, the iPad 'Settings' application reveals the IP address in use (usually acquired via DHCP) and allows the user to manually specify an IP address if needed. The IP address must be noted down, as it will be needed later for accessing the iPad from the remote computer.

Still in the 'Settings' application, the 'Auto-Lock' option must be set to 'Never'. This will prevent the device from going into sleep mode while the forensic image is being generated, which could interrupt the process. When not in use, the device should be locked (using the Lock button; see Section 2) in order to save battery.

We have not tested whether the multitasking capabilities and persistent Wi-Fi in iOS 4 would allow the imaging process to take place while the device is locked. Anyway, given that imaging is a process that can take a long time in devices with the largest local drives, it is recommended to keep the device awake all the time.

Local access approach

We would like to note that, even though the best way to apply our proposal is by issuing commands via SSH, we experimented and also found at least two ways to apply the imaging method without actually using a remote computer. However, both of them introduce additional complications to the process.

On one hand, it may be possible to install *MobileTerminal* instead of *openssh*, and use the terminal application in the iPad itself to mount the hard drive and image to it. However, at the time of writing, *MobileTerminal* does not work in iOS versions 4.x, and this software has a history of long delays before being updated to support newer iOS versions.

On the other hand, another approach is to install *openssh* and run an SSH client on the iPad itself. There are many such applications in Apple's App Store, although keeping this application running during the image generation is likely to alter data and will possibly corrupt the image. Thus, it is preferable to use a remote computer and leave the iPad as untouched as possible. Setting the user data partition read-only is not a possible solution in this case, as will be explained in Section 4.2.1.

Fig. 2. iPad connection to an external hard drive via Camera Connection Kit.

4.2. Device imaging

Before the process can actually begin, it is necessary to have the battery charged to, at least, about 20%. This is because, during the imaging process, the iPad's dock connector will be used for USB data transfer, so it will not be possible to plug the device to a power point.

With the device connected to a wireless network, another computer is used to establish an SSH session with the iPad. At this point, the device is accessible via the standard password `alpine`, which works for both the standard `mobile` user as well as for the `root` user, which has full access to the device. The correct way to proceed would be to access the iPad via SSH as the `root` user, and immediately change its password and the password of the `mobile` user account, using the `passwd` command.

4.2.1. Mounting a USB hard drive

In this step, the Apple peripheral, Camera Connection Kit [17], is used in order to access an external USB hard drive. According to Apple, "*the iPad Camera Connection Kit gives you two ways to import photos and videos from a digital camera: using your camera's USB cable or directly from an SD card*" [17]. Thus, it consists of two adapters, one of them being an SD card reader and the other offering a USB female connector; both of these adapters can plug (one at a time) into the iPad's dock connector, located in the base, below the "Home" button.

Initial vendor information suggested that the USB adapter only uses the PTP [18] protocol to access the images stored in a camera, and that an actual camera, with its camera-to-USB cable, should be plugged into this connector for the adapter to import the pictures. This is the normal use for this adapter, and is what Apple advertises it for. When this is done, the *Photo* application launches and allows the user to transfer photos and videos from the connected media to the iPad's internal memory.

Our results show that iOS implements the much wider *USB mass storage device class* protocol, of which PTP is a subset. PTP allows basic operations on the items of a video device, usually a photo camera, in which FAT-formatted media store the files under the `/DCIM` directory [19]). Instead, the iPad can mount and make full use of FAT and HFS filesystems.

The device just emulates the behavior of PTP: it automatically mounts FAT drives looking for the `/DCIM`. If this folder exists, the *Photo* application will open, allowing the user to import contents; if it does not exist, the device is unmounted and ignored. In our approach, we exploit this undocumented feature to manually mount an external USB hard drive with the appropriate parameters, and use it to obtain an image of the iPad internal storage.

After connecting the USB external drive to the iPad (see Fig. 2), we can check its presence within the SSH session by looking for the `/dev/disk1` device. The iPad internal storage disk is assigned the device name `/dev/disk0`, so the presence of such a device implies that the newly connected hard drive has been correctly recognized.

Should no `/dev/disk1` device exist, the drive has not been recognized. In our tests, this was usually accompanied by a dialog in the screen complaining that "this device requires too much power", when trying to connect certain large solid state storage drives and some portable hard drives that take power from USB only. Under iOS version 4 it is more problematic, since the USB port will no longer emit 100 mA (as it did under iOS 3.x) but only about 20 mA [20]. Therefore, in our tests, we found that the best results were achieved using a full-size external hard drive with its own power adapter, or connecting the drive to a powered USB hub.

As for the supported filesystems, we have been successful in mounting FAT and HFS + (the standard Macintosh filesystem, which is also the one used for the iPad internal storage). Due to the Macintosh EFI support, finding the correct partition name

for disks with EFI partitioning can be tricky. It is not always named `/dev/disk1`. In that scenario, we found it is necessary to list all the extra available disk devices and mount them, one at a time, until one is successful.

An important issue for Windows users is that their operating system will refuse to format a drive larger than 32 GB as FAT [21], although it can normally mount much bigger FAT partitions and work with them flawlessly. These users will need to use externals tools such as Fat32Format [22]. Mac and Linux users will have no trouble with their standard *Disk Utility* and *mkfs.msdos* tools, respectively.

Zdziarski [2] recommended immediately remounting the data partition in read-only mode prior to beginning the actual imaging. However, in our tests, we found that in both iOS 3 and iOS 4 the system halted if the partition was unmounted. Forcing its remount was not supported either. It must also be noted that imaging a mounted partition may alter the integrity of the filesystem contained in the resulting image. In fact we found out that it is possible to end up with images that are unmountable. In order to reduce this risk, no other activity should take place in the iPad (either via the touch screen or through the network) while imaging.

Considering all the above, the best option is to use a drive with traditional MBR partitioning scheme, containing only a HFS + partition. Once the disk has been mounted, it is possible to assess its free space and confirm that the drive has been correctly recognized.

4.2.2. Obtaining the forensic image

At this point, the actual imaging process may be started by invoking the Unix low-level copying and raw data conversion `dd` command, as listed in the following line. The resulting data is dumped to the newly mounted filesystem via the USB connection.

```
# dd if=/dev/rdisk0s2s1 of=/mnt/ipad.dd bs=512k
```

If the target drive is FAT32-formatted, the maximum file size it can hold will be 4 GB. This means that images will need to be split into 4 GB blocks. In the recommended scenario (imaging to HFS + drives), this is not necessary.

When finished, the target drive **must** be unmounted before disconnecting it from the iPad. This can be done by either turning the iPad off or unmounting the drive by exiting the `/mnt` folder and running the Unix `umount` command.

If the device is passcode-protected, jailbreaking and accessing via SSH is equally possible. The simplest jailbreaking methods from a user standpoint, such as `jailbreakme.com`, will not work given that we are unable to obtain initial access to the device, but other methods (*redsn0w*, *Pwnage Tool*…) may work.

5. Discussion on image analysis

Using our proposed open method, it is possible to obtain the data stored in the user partition in a manner that can be readily accessed using forensic tools and techniques. This method can be automatically applied to new iOS versions as soon as a jailbreak is available for them, even if it is just a *tethered* jailbreak.

In this section, we provide a brief discussion about how the resulting image can be processed and what kinds of data can be obtained from our image, highlighting some important issues in more recent iOS revisions in that regard.

5.1. Obtainable data using forensic tools

Raw images obtained by our method can be treated in many ways. For instance:

- SQLite databases, which can be open with a variety of SQLite clients available for all platforms, will show information such as address book contacts, call history, SMS and e-mail messages, etc. This includes the `consolidated.db` file, which in some iOS 4.x versions keeps a complete log of the user location forever, which can be parsed by iPhoneTracker [23].
- Property list (`.plist`) files containing XML information can be viewed with tools available for all platforms. These files contain other relevant data such as: website bookmarks and cookies, maps history, as well as configuration information for each application installed.
- Images and videos can be viewed. Moreover, each of them is likely to contain embedded GPS coordinates of the place where it was taken. This can be very relevant to certain investigations.
- And finally, as a wise forensic examiner once said, 'when all else fails, we carve'. Raw disk recovery (file carving) tools can be used to recover any file type based on its signature: pictures, voice memos, raw text…This is an extreme resource that implies getting many corrupt files and disclosing valuable data such as names, paths, and timestamps; however it is still a very valuable technique, helpful in many scenarios. Open source tools such as Scalpel [24], as well as most commercial data recovery tools, can be used to extract raw data from the image.

We verified that these and other valuable data can be extracted from an image generated by our method. All these possibilities are widely covered by Zdziarski [2] and Morrissey [25].

5.2. iOS 4 data encryption

Starting with iOS version 4, Apple introduced a layer of hardware encryption services called *data protection* [26]. This feature, if activated, will result in partial encryption of the imaged data. Therefore, it is a very important aspect as far as forensic image processing is concerned.

We tested the proposed imaging method on devices with this feature activated and experimented with the encryption layer, observing a new `protect` option for the `mount` command. Under a fresh installation (and subsequent jailbreaking) of iOS 4.2.1 on an iPad, running the `mount` command will show that this option is used for the data partition. The same happens on an iPhone 4. However, on an iPhone 3G (which does not support this feature), the `protect` option does not show up.

In addition, we have observed that iOS 4.2.1 `/sbin/mount` binaries in the iPad and iPhone 4 contain the *protect* string, but not in the iPhone 3G binary. The string is also present in several of the `/sbin/mount_*` binaries under Mac OS X 10.6 Snow Leopard, whereas those same binaries in Linux systems do not contain the '*protect*' string. This leads us to consider that, probably, the `protect` parameter for the `mount` command refers to the new *data protection* (local encryption) iOS feature.

There are two factors that weaken this encryption scheme:

(1) All original iPads sold between March and November 2010 (that is 8–10 million devices) shipped with iOS version 3.x. Even when undergoing a normal upgrade process to iOS 4.x, encryption will not be activated: for this to happen, the device must go through a full restoration, which requires completely restoring its data from the iTunes backup. This takes a notably longer time (one to two hours instead of barely 5–10 min) and will only happen if a fatal error rendered the device unbootable.
(2) Even if data protection is enabled, it could be circumvented as long as the encryption keys are stored within the device itself.

In order to determine whether the encryption keys are stored in the device (i.e. the device can mount the encrypted data automatically, without the need of any user input or network connectivity), we performed the following checks:

(1) The following test was conducted over two devices: a first-generation iPad and an iPhone 4, both of them running iOS version 4.3.1.
(2) The device was jailbroken and an SSH server installed.
(3) Using the iOS 'Settings' app, GSM/3G data connectivity was disabled. In addition, 'flight mode' was enabled.
(4) Wi-Fi connectivity was then enabled again. This causes the device to remain in 'flight mode' (absolutely no GSM/3G activity) while allowing for 802.11 wireless communication.
(5) The device was connected to a Wi-Fi network with no Internet connectivity. Other nearby networks stored in the device's 'preferred networks' list were erased, to ensure that upon reboot the device would join our test network.
(6) The device was turned off, and then on, booting normally.
(7) Once the boot process finished, the device would show its lock screen, asking for the passcode. At this point, the device was left untouched.
(8) From a different computer in the same Wi-Fi network, we initiated a SSH connection to the device, running the 'mount' command and confirming that the data partition (`/dev/disk0s2s1`) was already mounted under `/private/var`.

The encrypted volume is always automatically mounted with no need for passcode entry or Internet connectivity. This leads us to the conclusion that iOS 4 local encryption relies on a key stored within the device itself, which implies that forensic images acquired through our method can be decrypted—even when certain data may be additionally encrypted with the user passcode, which seems to be the case with the Mail application and others.

In fact, in May 2011, ElcomSoft announced [27] the breaking of iOS 4 encryption, with a smart approximation that can extract most encryption keys from the physical device, which confirms that the keys reside within the device itself. However, ElcomSoft restricts the availability of the toolkit to select government entities such as law enforcement and forensic organizations and intelligence agencies.

5.3. Obtaining relevant information from encrypted images

Even though, to our knowledge, no open and easily available mechanism exists to defeat iOS 4 encryption, our experiments have shown that, nevertheless, it is possible to obtain some data from our image. Even though encrypted images initially fail to mount under Mac OS X, we were able to find that, surprisingly, when those images are truncated (i.e. the filesystem loses its final bytes), they mount correctly. These results persisted in all our tests, in which we shortened the images by trailing the final bytes (from 1 KB to 1 GB). We did not run further tests because larger cuts would, in the end, generate problems when reading the images. However, this behavior seems to indicate that the 'encrypted' flag resides in a header at the end of the filesystem and, when this header is missing, OS X does its best to mount the filesystem anyway.

Using this mechanism, the full list of files and directories can be obtained, as well as other details such as last modification time or file length. The file contents themselves are encrypted, but a lot of useful information about the device usage can still be obtained this way, such as the list of installed applications or their last usage date.

In many cases, just the existence of files with certain names and sizes can be relevant evidence. For instance, consider a corporate information leak: if an iOS network client was used to extract the leaked files, or even if just some of the leaked files have been read in an iOS application, the image will show files with:

- Names, sizes, and *m*-time (last Modification time) matching those of the original, leaked files.

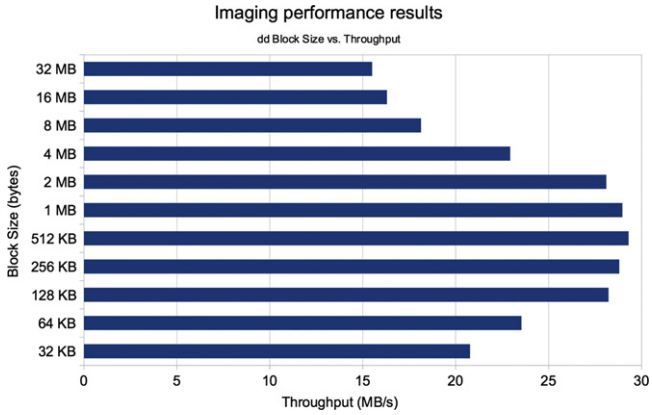

Fig. 3. Throughput analysis with different block sizes.

Table 1
Test results: throughput obtained with different block sizes.

bs	S1 (MB/s)	S2 (MB/s)	S3 (MB/s)	S (MB/s)
32 KB	20.7	20.8	20.8	20.77
64 KB	20.7	25.1	25.1	23.54
128 KB	29.5	27.6	27.6	28.22
256 KB	28.8	28.8	28.8	28.80
512 KB	28.8	29.5	29.6	29.30
1 MB	29.0	28.9	29.0	28.97
2 MB	28.1	28.1	28.1	28.10
4 MB	22.9	22.9	23.0	22.93
8 MB	18.1	18.1	18.2	18.13
16 MB	16.3	16.3	16.3	16.30
32 MB	15.5	15.5	15.5	15.50

- *c*-time (Creation time, or time of the last status Change), indicating when the files first arrived on this device (or, if they were moved to a different directory afterwards, indicating when that happened).
- *a*-time (last Access time), indicating when the files were last read, which can indicate whether the files have been manually reviewed by the user after downloading them.
- Their allocation under the folder of the corresponding iOS application that generated them.

6. Performance analysis

The most important parameter in the efficiency of our proposed method, as far as imaging speed is concerned, is the `bs` (*block size*). In order to obtain an optimal value, we ran a series of tests for each value. External interference was minimized by turning off Bluetooth, 3G data, push notifications, push e-mail, and localization services. Wireless connectivity was needed for our tests, however our wireless network did not have a gateway to the Internet, again to keep the device activity to a minimum. After these settings were applied, the device was rebooted into a clean state.

Under these conditions we ran three complete 63.5 GB dumps for 11 different values for `bs`, amounting to a total of 1.90 terabytes in 33 dumps. The speed obtained in each test (S1, S2, S3) can be seen in detail in the corresponding Table 1 (see Fig. 3 for a quick summary).

We were surprised to find that small values (128 KB–2 MB) provide the best throughput, whereas larger or smaller values result in a notable speed decrease. This is in contrast to what happens with traditional computers (be they desktops or laptops), in which the best results are obtained with block sizes of 16 or 32 MB, usually matching typical hard drive cache sizes.

In comparison, we tested the Zdziarski method over an ad-hoc 802.11n network operating at its maximum theoretical rate (108 Mbps), and we obtained a throughput of barely 1 MB/s, which means that a 16 GB iPad would be imaged in 5 h and a 64 GB one would require about 20 h. Our USB approach results in a speed boost of nearly 30× over traditional Wi-Fi imaging.

Forensics software vendors do not seem to release specifications about the imaging times needed by their methods; we could only find that information for Jonathan Zdziarski, who states [28] "*the latest version of the Zdziarski method, which is used in the automated tools available free to law enforcement agencies worldwide*": "*about* 15–30 min *is all it takes, regardless of whether you're imaging a* 4 GB *iPhone or a* 32 GB *iPhone* 3*G[s]*". Assuming he is able to image 32 GB in 'about 30 min', our method is twice as fast. However, Zdziarski does not mention how much time he needs to image an iPad, and our

method cannot be tested in an iPhone because the iPhone does not support the Camera Connection Kit. Thus, the different throughputs could also indicate that the iPad's serial port is faster than that of the iPhone.

7. Conclusions and future work

In this paper, we have presented a novel approach that takes advantage of a hidden feature in the iPad's USB adapter so it is possible to use a cheap, universally available $ 30 accessory to image the device directly to a USB drive attached to it. The main contribution of this approach is in providing a mechanism that is fast, open and easily available to anyone, and that automatically keeps working regardless of the iOS version.

To date, high transfer rates could only be achieved using commercial tools which are paid for and/or restricted to law enforcement agencies, opaque to the scientific community and undocumented. Therefore, it is difficult to assess what is really happening during the imaging process and whether the original data is somehow being altered. Furthermore, they only work for specific iOS versions, which becomes a hassle when the current iOS is upgraded. In fact, it may be the case that, for some time, no such tool is available.

On the other hand, open and easily upgradeable approaches are entirely based on Wi-Fi image transfer, being very slow. Our approach offers what appears to be the best of both worlds, becoming one of the fastest ways to obtain a complete forensic dump of the Apple iPad with these characteristics. In fact, we have apparently reached the speed limit of the iPad's dock connector. This is accomplished using a generic UNIX approach via jailbreaking, which is likely to last longer than most iPad forensics software products, thus guaranteeing that it can be applied in future iOS versions.

As far as further work is concerned, currently, the main challenge to the whole process of iPad forensic imaging is the new encryption layer for iOS 4. Even though the device *data protect* mode is not widespread yet and, since very recently, law enforcement agencies have methods at their disposal that break such encryption, we deem it interesting to perform an in-depth study of the iOS 4 encryption system and make further analysis of a protected image.

Nevertheless, there is still some interesting work that does not need a full disk image, such as being able to perform a live memory dump of the device, autonomous imaging from the iPad to the connected USB drive, eliminating the need for a network and a remote computer, and analyzing the forensic artifacts left by the AirPrint subsystem in the device's local disk.

Acknowledgments

This work was partially supported by the Spanish MCYT and the FEDER funds under grant TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 "ARES", funded by the Spanish Ministry of Science and Education.

References

[1] InformationWeek, iPad is top selling tech gadget ever, 2010. http://www.informationweek.com/showArticle.jhtml?articleID=227700347.
[2] J. Zdziarski, iPhone Forensics: Recovering Evidence, Personal Data, and Corporate Assets, O'Reilly & Associates Inc., 2008.
[3] Quickoffice Inc., Quickoffice connect mobile suite for iPad on the iTunes App, 2010. http://itunes.apple.com/us/app/quickoffice-connect-mobile/id376212724.
[4] Apple Computer Inc., Pages for iPad on the iTunes App. Store, 2010. http://itunes.apple.com/us/app/pages/id361309726.
[5] Apple Computer Inc., Apple's AirPrint Wireless Printing for iPad, iPhone and iPod Touch Coming to Users in November, 2010. http://www.apple.com/pr/library/2010/09/15airprint.html.
[6] EuroSmartz Ltd., PrintCentral for iPad on the iTunes App Store, 2010. http://itunes.apple.com/us/app/printcentral-for-ipad/id366020849.
[7] G. Hotz, iPhone serial hacked, full interactive shell, 2007. http://www.hackint0sh.org/f127/1408.htm.
[8] J.R. Rabaiotti, C.J. Hargreaves, Using a software exploit to image RAM on an embedded system, Digital Investigation 6 (2010) 95–103.
[9] Katana Forensics, Lantern, 2010. http://katanaforensics.com/use-our-tools/lantern.
[10] Forensic Telecommunications Services Ltd., iXAM-Advanced iPhone Forensics Imaging Software, 2010. http://www.ixam-forensics.com.
[11] L. Moenner, W. Jansen, A. Delaitre, Overcoming impediments to cell phone forensics, in: Proceedings of the 41st Annual Hawaii International Conference on System Sciences, IEEE CSP, 2008, pp. 483–483.
[12] Comex, Comex 'star' GIT repository, 2010. http://github.com/comex/star.
[13] Jailbreak Matrix, 2010. http://www.jailbreakmatrix.com/iPhone-iTouch-Jailbreak.
[14] iPhone Dev Team, Thanksgiving with Apple, 2010. http://blog.iphone-dev.org/post/1652053923/thanksgiving-with-apple.
[15] J. Freeman, Bringing Debian APT to the iPhone, 2008. http://www.saurik.com/id/1.
[16] Sophos Security, First iPhone worm discovered—ikee changes wallpaper to Rick Astley photo, 2009. http://nakedsecurity.sophos.com/2009/11/08/iphone-worm-discovered-wallpaper-rick-astley-photo.
[17] Apple Computer Inc, Apple iPad Camera Connection Kit, 2010. http://store.apple.com/us/product/MC531ZM/A.
[18] International Organization for Standardization (ISO), ISO 15740:2008—Electronic still picture imaging—picture transfer protocol (PTP) for digital still photography devices, 2008.
[19] Camera & Imaging Products Association, Design rule for camera file system: DCF version 2.0, 2010.
[20] 9to5 Mac, iOS 4.2 emits less USB power on iPad, Camera Connection Kit crippled?, 2010. http://www.9to5mac.com/40091/ios-4-2-emits-less-usb-power-on-ipad-camera-connection-kit-crippled.
[21] Microsoft Corp., Limitations of the FAT32 file system in windows XP, 2007. http://support.microsoft.com/kb/314463.
[22] Ridgecrop Consultants Ltd., Fat32Format, 2009. http://www.ridgecrop.demon.co.uk/index.htm?fat32format.htm.
[23] Alasdair Allan & Pete Warden, iPhone Tracker, 2011. http://petewarden.github.com/iPhoneTracker/.
[24] Digital Forensics Solutions, Scalpel, 2011. http://www.digitalforensicssolutions.com/Scalpel/.
[25] Sean Morrissey. iOS forensic analysis for iPhone, iPad, and iPod touch. apress, 2010.
[26] Apple Computer Inc, iOS 4: understanding data protection, 2010. http://support.apple.com/kb/HT4175.
[27] ElcomSoft Co. Ltd., Enhanced forensic access to iPhone/iPad/iPod devices running iOS 4, 2011. http://www.elcomsoft.com/iphone-forensic-toolkit.html.
[28] J. Zdziarski, iPhone insecurity, 2010. http://www.iphoneinsecurity.com.